

SMW Ontologies & Variables

Building on SMW 2.x

John McClure
BellTowerWikis

jmcclure@hypergrove.com

Warning: Firehose ahead !

Presentation Topics

Enhancements for SMW

1. Initial lower-cased Property pages
2. MyLanguage links
3. Filtered inverse relations
4. Localized property values
5. Localized query results
6. User query variables

Establish a community ontology

1. Base SMW ontology
2. Legacy ontologies
3. Grammatical ontologies

Initial lower-cased Property: pages

SMW does not honor \$wgCapitalLinks

“Rdf:type” is simply *not* the same as “rdf:type”

At a minimum is aesthetically displeasing

Change to SMW's global smwgNormalizeTitle()

Tested and works fine

SMW Localization

- Without enhancement SMW is effectively not usable by multi-lingual wikis
- Only approach available now is to define language-specific properties, e.g.,
[[dc:title :: some english]]
[[dc:title (fr) :: some french]]
- Page templates are complex to write
- Performance is less than optimal
- New wiki language addition is not transparent
- “Property:dc:title (fr)” defeats interchange

MyLanguage Links

```
{{#ask: [[predicate]]  
| link = all || none || subject || local || local-subject  
}}
```

-
- Formats links with prefix “Special:MyLanguage/”

Should the link then be clicked by a user,
Special:MyLanguage looks up user's language
and displays the correct subpage if it exists

- “local” operates on all links in a query result
- “local-subject” operates on subject links only

Localized Property Labels

Must standardize method for localizing property names
– property names needed by numerous Special pages

Direct language-named annotations seem most efficient

Property:*pagename*

[[Has type :: Text]]
[[@en :: *english label*]]
[[@fr :: *french label*]]

Other than display, labels are irrelevant to all SMW operations
e.g., [[*french label* :: *value*]] refers to an undefined property
Rather, the label is only used for display text in a link to the property.

Localized Property Values

On page “Movie: Blazing Saddles”, assume

Text properties:

[[dc:title @en::Blazing Saddles]]

[[dc:title @fr :: Brûlant Selles]]

Page properties:

[[Has advertisement @en :: Ad: Blazing Saddles]]

[[Has advertisement @fr :: Ad: Brûlant Selles]]

-
- Conforms to xml:lang recs (RDF/Turtle & Wikidata JSON)
 - **No** semantic markup is placed onto a page's localized subpages
 - e.g., Movie: Blazing Saddles vs Movie:Blazing Saddles/fr
 - only [[Movie: Blazing Saddles]] sets page properties

Localized Property Values

```
{{#set: dc:title=Blazing Saddles  
| dc:title @fr=Brûlant Selles  
}}
```

Stored as

```
{{#subobject: dc:title  
| @en=Blazing Saddles  
| @fr=Brûlant Selles  
| xp:parent=Movie: Blazing Saddles  
}}
```


Localized Query Results

```
{{#ask: [[ Movie: Blazing Saddles ]]  
| ?dc:title  
| ?Has advertisement  
}}
```

-
- Query returns value(s) for current UI language
 - Just two properties (dc:title & Has advertisement)

Localized Query Results

```
{{#ask: [[ Movie: Blazing Saddles ]]  
| ?dc:title @en  
| ?Has advertisement @en = English Advert  
| ?Has advertisement @fr = French Advert  
}}
```

The query returns all property values tagged with requested language(s)

Localized Query Results

```
{{#ask: [[ Movie: Blazing Saddles ]]  
| ?dc:title = @en:Movie; @fr:Film  
}}
```

-
- Prints column label per current UI language
 - Value is that which defaults to wiki's language
 - Last chance: prints first label provided
 - No attempt to resolve to an 'affinity' language
 - Alternative (requires page “MediaWiki:Movie label”):

```
{{#ask: [[Movie: Blazing Saddles]]  
| ?dc:title = {{int:Movie label}}  
}}
```

Localized Query Results

```
{{#ask: [[ Movie: Blazing Saddles ]]  
| ?dc:title = @en:Movie; @fr:Film  
| @=fr  
}}
```

-
- Queries as if all components qualified with @fr
 - Simplifies templated queries and Special:Ask

Localized Query Results

A predicate such as

`[[dc:title :: Blazing Saddles]]`

is the same as

`[[dc:title @wiki-language :: Blazing Saddles]]`

A predicate such as

`[[dc:title :: @fr:Brûlant Selles]]`

is the same as

`[[dc:title @fr :: Brûlant Selles]]`

Therefore a predicate such as

`[[dc:title @fr :: Blazing Saddles]]`

returns null.

Filtered Inverse Relations

Assume on page “Michael”:

[[Course enrollment::English]]

[[Course enrollment::Biology]]

Possible:

{{#ask: [[-Course enrollment::Michael]] }}

Returns all courses in which Michael is enrolled
i.e., Biology Course and English Course

Not possible:

[[[-Course enrollment::Michael]] [[Category:Science courses]]

Possible but very unsatisfying:

```
{{#ask: [[-Course enrollment::Michael]]  
  | template = filterForScienceCourses  
  }}
```

2nd recourse: {{#ask: [[-Course enrollment::Michael]] |format=array}}

User Query Variables

MySQL Select Statement

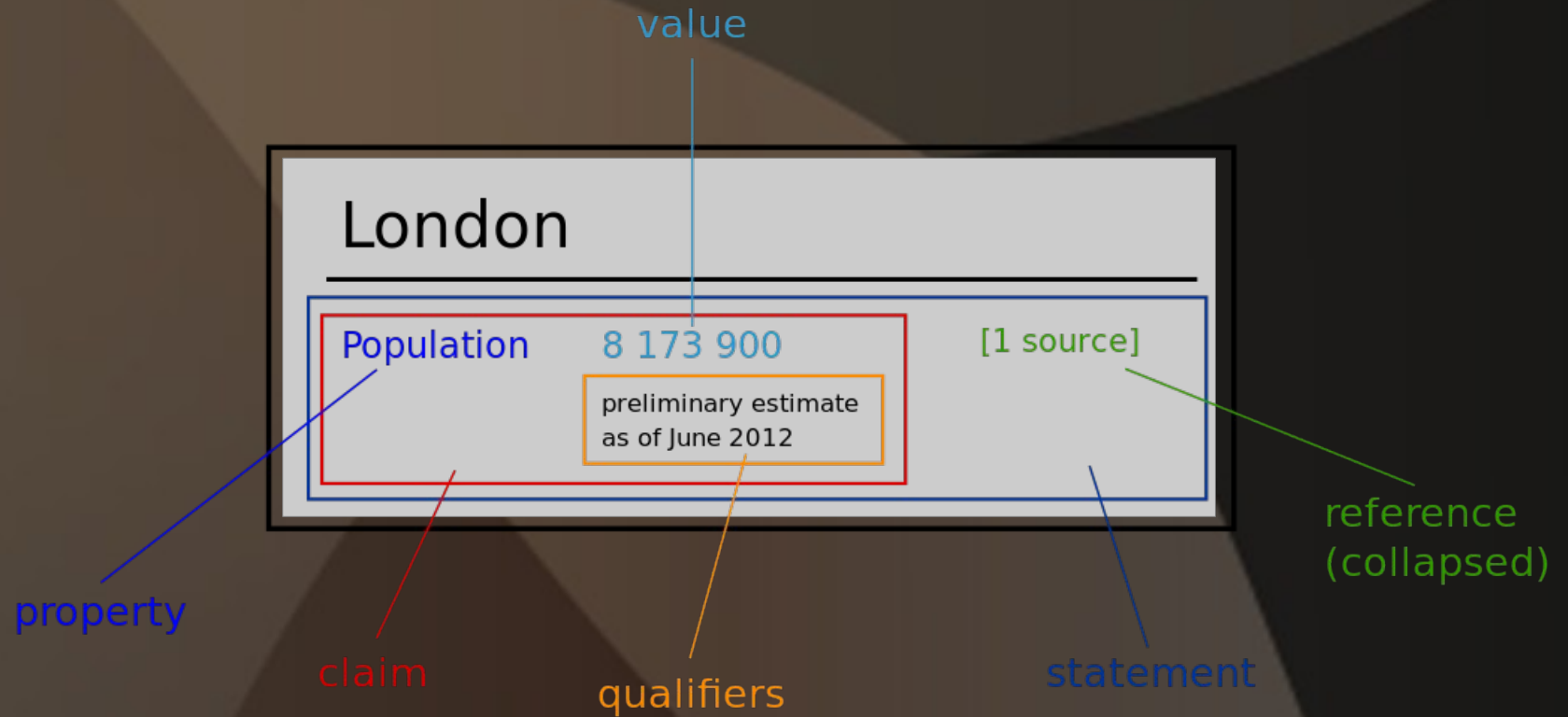
```
@num = 0; @type = "";  
select type, variety, price, @num  
from fruits  
where 2 >= greatest(  
    @num := if(@type = type, @num + 1, 1),  
    least(0, length(@type := type)))  
order by type, price;
```

SMW Equivalent?

```
{{#ask: [[ rdf:id ::~ Fruit: * ]]  
| ?type |?variety |?price  
|template=calculate_num  
|sort=type, price  
|order=ascending  
}}
```

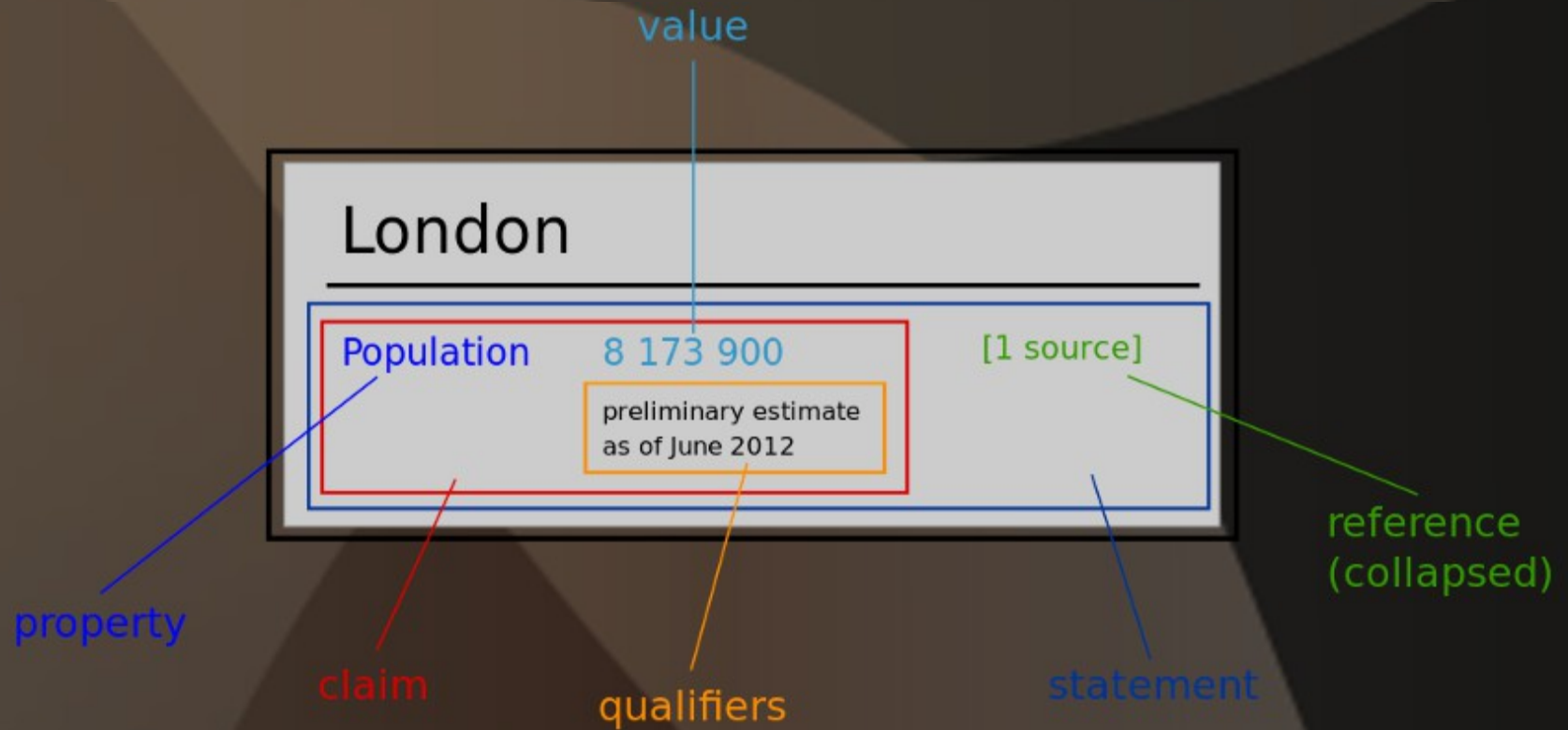
NO! Template:calculate_num
cannot be written!

Start with an “ABox”



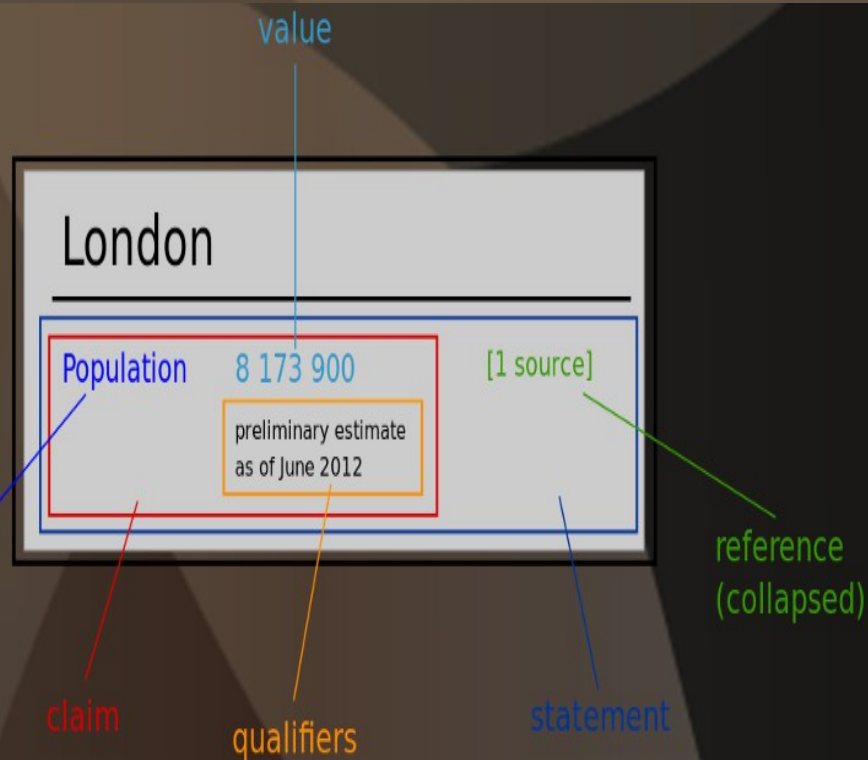
Source: Wikidata.org

Start with an “ABox”



Source: Wikidata.org

Turtle Representation



```
this:topic
  dc:title      "City: London UK";
  has:this      #Census: 1995 Residents;
.
#Census: Residents (1995), London UK
  be:about      this:topic;
  is:for        Demographic: Residents;
  is:for        Year: 1995;
  xsd:integer   8173900;
  dc:date       "June, 2012";
  dc:source      "statement of source";
  dc:Source      Source: some-source;
  dc:Type        Tag: Estimated;
  dc:Type        Tag: Preliminary;
.
[1] this:topic is a reserved subject
```

SMW Query Variables

SMW Query with variables

```
{{#ask: [[City: London UK]]
|*census=[[-has:this]][[Type:Census]]
|**value  = /.xsd:integer
|**year   = [[-is:for]][[Type: Year]]
|**source = /.dc:source
|**date   = /.dc:date
|**quals  = /.dc:Type
|?value   = @en:Value; @es:Valor
|?year    = @en:Year
|?quals/.smw:PAGENAME=@en:Qualifiers
|?source  = @en:Source
|?date    = @en:Date
|link     = local
|format   = template
|template = wikibase-box
|intro    = <div class='myclass'>
|outro    = </div>
}}
```

Notes about the query parameters

Retrieve the London UK resource
retrieve Census resource (new scope)
'/.xsd:integer' retrieves value
retrieve Year resource
call template/parser-fn
..same
..same
note multilingual column label
..same
call .smw:PAGENAME template/pfn
..same
..same
prefix links with Special:MyLanguage/
pass each row (6 items) to a template
what template to pass row data to
what text to first put into output
what text to append to output

SMW Query Variables permit complex logic

Scenario: Determine William Shakespeare's birthdate.

A date of birth can be specified more than one way...

1. Directly upon Shakespeare's page
 - a) dc:date "date"
 2. Within a "Date" object/page related to WS's page by either a "is:from" or "was:from" property
 3. Within a "Date" object/page related to a "Birth" object/page itself related to WS's page by either "has:this" or "had:this"
 - a) is:after or is:before or is:circa a date
 - b) is:during a period of time
 - c) is:on or was:on a date
-

Query variables syntax:

- |* sets variable (**continue** query processing)
- |*? sets variable (**end** query processing if non-null)
- |*var-name.integer sets nth value of variable

Query for date of birth:

```
{#ask: [[Person: William Shakespeare]]
|*from = [[-is:from]][[Type: Date]] OR [[-was:from]][[Type: Date]]
|**source = /.dc:source
|**tags    = "n/a"
|*?date    = /.skos:prefLabel
|*birth    = [[-has:this]][[Type: Birth]] OR [[-had:this]][[Type: Birth]]
```

```
|**bdate = [[-is:on]][[Type: Date]] OR [[-was:on]][[Type: Date]] OR
           [[-is:this]][[Type: Date]] OR [[-was:this]][[Type: Date]]
|***source = /.dc:source
|***tags   = /.dc:type
|**?date   = /.skos:prefLabel
|**?date   = /.xsd:date
```

```
|**after   = [[-is:after]][[Type: Date]]
|***source = /.dc:source
|***tags   = /.dc:type
|***date.1 = @en:after;@es:despues
|**?date.2 = /.skos:prefLabel
|**before  = [[-is:before]][[Type: Date]]
|***source = /.dc:source
|***tags   = /.dc:type
|***date.1 = @en:before;@es:antes
|**?date.2 = /.skos:prefLabel
|**circa   = [[-is:circa]][[Type: Date]]
|***source = /.dc:source
|***tags   = /.dc:type
|***date.1 = @en:circa;@es:hacia
|**?date.2 = /.skos:prefLabel
```

```
|**during   = [[-is:during]]
              [[Type:Timespan]]
|***source = /.dc:source
|***tags   = /.dc:type
|***date.1 = @en:during;@es:durante
|**?date.2 = /.skos:prefLabel
|**source  = n/a
|**tags    = n/a
|**date     = n/a
```

```
|?date= @en:Birthdate;@es:Cumpleanos
|?source = @en:Source;@es:Fuente
|?tags   = @en:Tags; @es:Etiqueta
```

```
|format = template
|template = .wd:statement
}}
```

SMW Ontologies

A Base Ontology

- No intention to standardize – sharing experience
 - affects ability to share/extend forms & queries
- SMW properties are part of an “smw” namespace
 - permits leveraging general ontology architecture
 - distinguishes “built-in” versus user properties
 - alternatives: #redirect vs PHP globals changes
- SMW defines no classes (“categories” or “concepts”?)
 - are SMW's “types”, classes or properties?
 - They're both!
 - They're each actually a “Datatype”

SMW Ontologies

A Base Ontology

Classes

smw:Type
SMW extensions' classes

Special:Types

smw:Annotation URI
smw:Boolean
smw:Code
smw>Date
smw:Email
smw:Coordinates
smw:Number
smw:Page
smw:Quantity
smw:Record
smw:Telephone
smw:Temperature
smw:Text
smw:URL

Conventions:
lower-case property names
upper-case class names

A basic
relation property

The most basic
attribute property

Properties

smw:type
smw:special properties
extra special properties
SMW extensions' properties

Special:Types

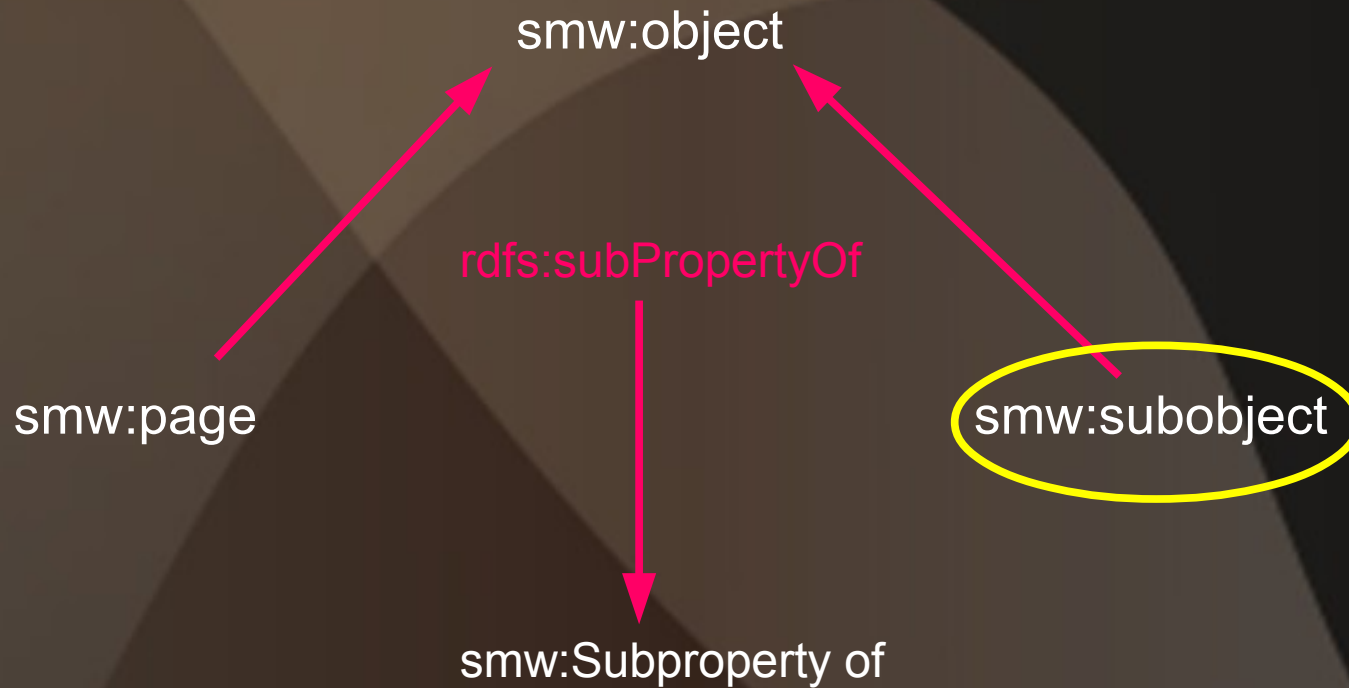
smw:annotationURI
smw:boolean
smw:code
smw:date
smw:email
smw:coordinates
smw:number
smw:page
smw:quantity
smw:record
smw:telephone
smw:temperature
smw:text
smw:url

What's
missing
in this
list?

SMW Ontologies

A Base Ontology

What's
missing in
SMW's list
of types?



Note:

`smw:subobject == smw:Has subobject`

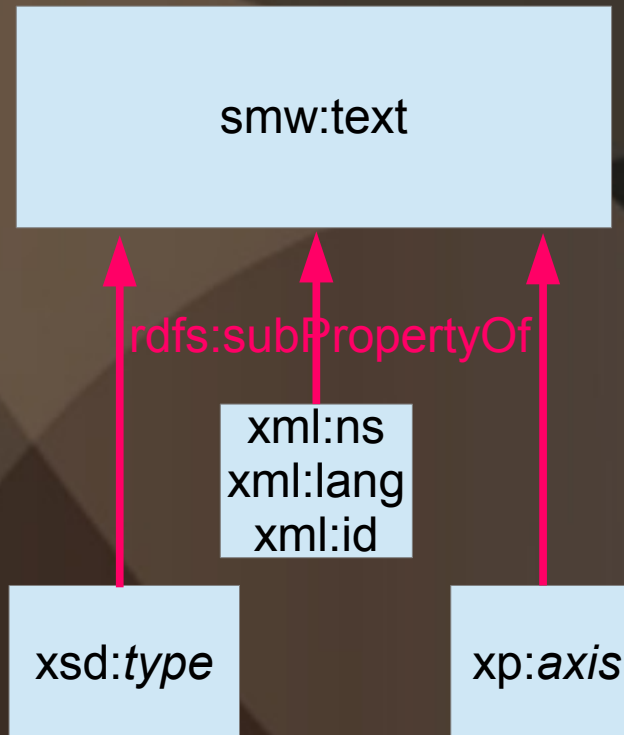
A Base SMW Ontology

XML-related Properties

Basic text properties

XSD Primitives

xsd:string
xsd:boolean
xsd:decimal
xsd:float
xsd:double
xsd:duration
xsd:dateTime
xsd:time
xsd:date
xsd:gYearMonth
xsd:gYear
xsd:gMonthDay
xsd:gDay
xsd:gMonth
xsd:hexBinary
xsd:base64Binary
xsd:anyURI
xsd:qname
xsd:notation



Basic relation properties

Xpath Axes

xp:ancestor
xp:ancestor-or-self
xp:attribute
xp:child
xp:descendant
xp:descendant-or-self
xp:following
xp:following-sibling
xp:namespace
xp:parent
xp:preceding-parent
xp:preceding-sibling
xp:self

#redirect [[Property:property]]
is used for certain “same as” properties

SMW Ontologies

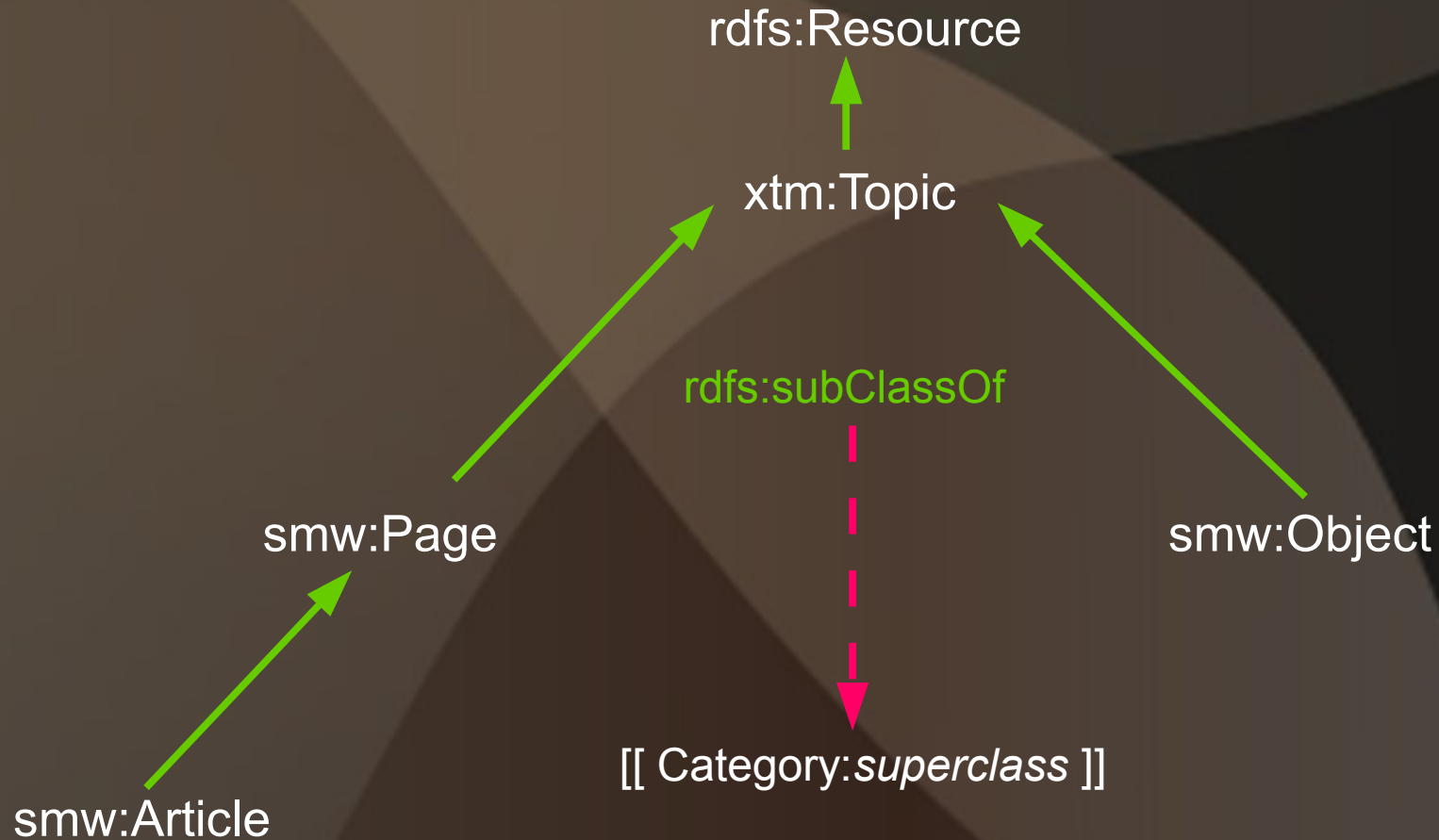
Why is all this subPropertyOf stuff useful?

End the “empty wiki syndrome”

- Property storage templates reuse super-property's storage templates
- Adheres to fundamental software engineering principles
- “Out of the box” re-useable components for user's templates & properties
- Adopting well-known ontologies benefits interoperability enormously
- Two types of templates are actually at issue
 - Data storage templates (semantic templates) eg `{{rdfs:subPropertyOf}}`
 - Data display templates (dotted templates) eg `{{.rdfs:subPropertyOf}}`
- Conversion to Lua modules for performance improvements
- Templates used in queries involving SMW variables

SMW Ontologies

Base Classes



There's an obvious relationship
between nouns & classes

SMW Ontologies

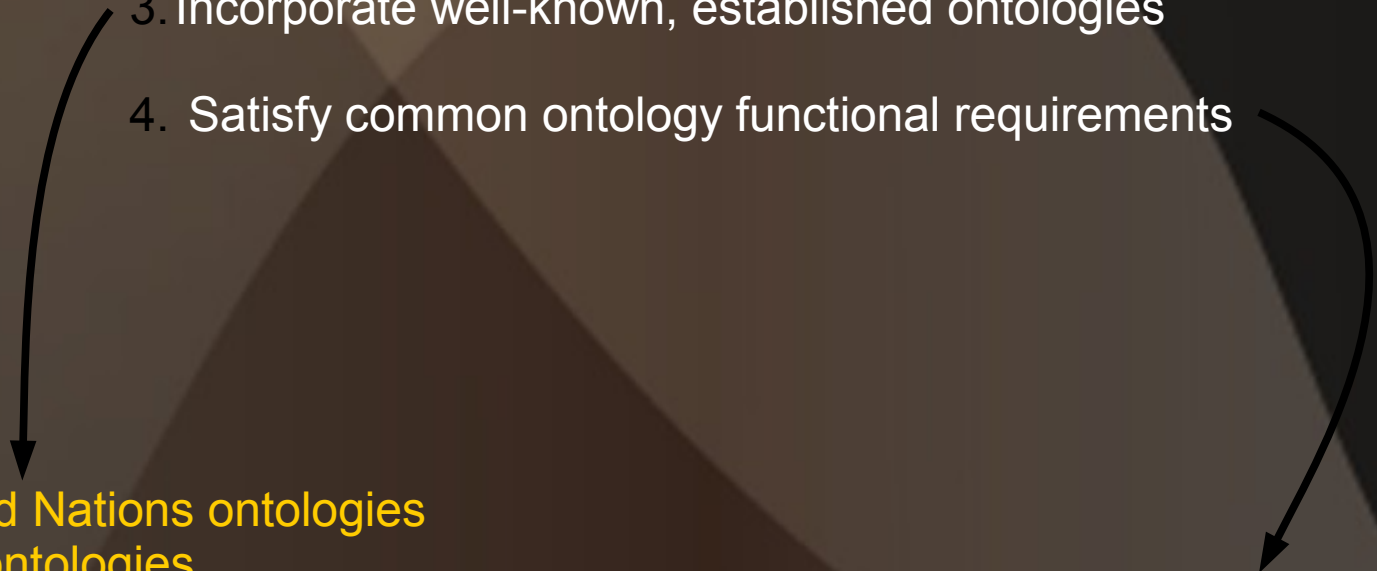
Five Architectural Design Rules

1. Specific nouns are classes containing instances of those nouns
e.g., `[[Category:Car]]`
2. Noun modifiers – adjectives and (past) participals – are classes containing instances of things with that characteristic
e.g., `[[Tag:Red]]` `[[Tag:Received]]`
3. Verb modifiers – adverbs – are classes containing instances of relations with that characteristic
e.g., `[[Tag:Temporarily]]`
4. Prepositions & indicatives (determiners) are properties that define specific relations between subject/object nouns
e.g., `San Francisco – in – California`
5. Verbs are (relation) property namespaces
e.g., `San Francisco – is:in – California`
`San Francisco – is-not:in – Nevada`

SMW Ontologies

Architectural Objectives

1. Mimic & validate USERS' cognitive model
2. Fixed number of properties but open-ended number of class definitions
3. Incorporate well-known, established ontologies
4. Satisfy common ontology functional requirements

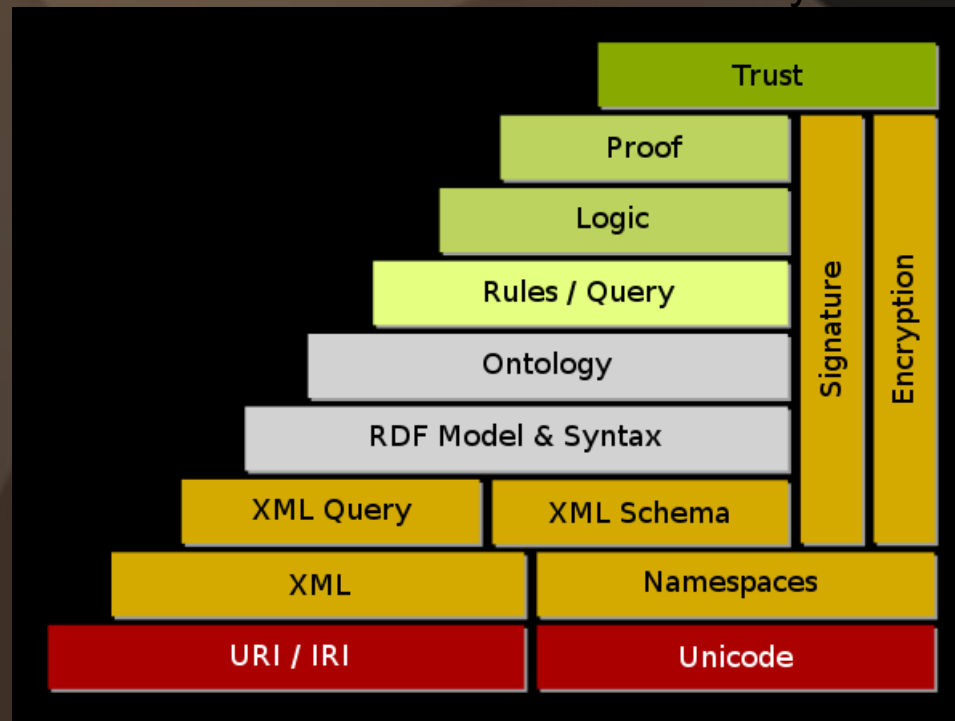
- 
- United Nations ontologies
 - ISO ontologies
 - W3 ontologies
 - OMG ontologies
 - Others e.g., GoodRelations

<http://ontologydesignpatterns.org>

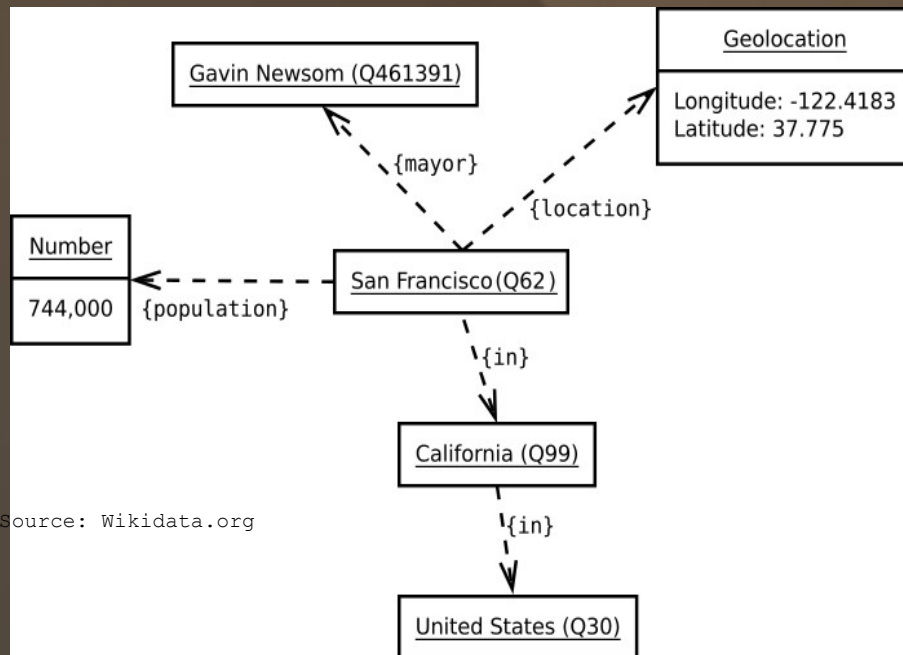
SMW Ontologies

Thank you!

The semantic web – architectural layers



SMW Ontologies



```

this:topic
  dc:title      "San Francisco CA" @en;
  rdf:type      Type: City;
  geo:lat       37.775;
  geo:lon       -122.4183;
  has:this      #Census: San Francisco CA;
.
#Census: San Francisco CA
  be:about      this:topic;
  xsd:integer   744000;
.
#Property: geo:lat
  rdfs:subPropertyOf Property: geo:lat;
  rdfs:label        "Latitude" @en;
  dc:Type            Tag:Precise;
.
#Property: geo:lon
  rdfs:subPropertyOf Property: geo:lon;
  rdfs:label        "Longitude" @en;
  dc:Type            Tag:Precise;
.
  
```